REMARKS/ARGUMENT

Claims 1-21 are pending in the application. Claim 1 is rejected under 35 U.S.C. §101. Claim 1 is rejected under 35 U.S.C. §102(a) as being anticipated by Gottlieb (U.S. Patent No. 6,016,542). Claims 1-21 are rejected under 35 U.S.C. §102(e) as being anticipated by Eikemeyer (U.S. Patent No. 6,694,425).

With regard to the §101 rejection of claim 1, the Office Action alleges the practical application of determining if a stall is due to loading of memory and flushing of an instruction after a predetermined number of cycles if data is to be loaded from memory before execution is not clear. It further asserts the aforementioned limitations are a recitation of intended result, not a positive limitation. Finally, it asserts the focus is not on the step taken to achieve a final result which is useful, tangible, and concrete, but rather the final result achieved which is useful, tangible, and concrete. *See* Office Action dated 3/13/2007, paragraph 1. Applicants disagree.

With regard to the practical applications of the embodiment described in claim 1, Applicants submit support may be found at least at page 3, lines 9-16 of the specification. In this section, the need for a method to alleviate the slowing effects of stall in a multi-threaded execution pipeline in a processing system is discussed such that one of ordinary skill in the art would understand the practical application of the embodiment of claim 1. Applicants submit the current §101 rejection should be withdrawn.

With regard to the Office Action's assertions regarding a "positive limitation", Applicants submit claim 1 recites and further define steps of "determining" and "flushing", both of which constitute positive limitations in the context of a method claim applied with respect to memory and a processing system. With regard to the Office

Action's assertion that no predictable results can be found in the claim if the stalled operation is not due or when the data is not to be loaded before execution, Applicants submit the embodiment method of claim 1, for example, positively describes the method in the context if the stalled operation is due and when the data is to be loaded before execution. No further clarification is necessary. Applicants maintain the current §101 rejection should be withdrawn.

With regard to the Office Action's assertion regarding the focus of the final result is useful, tangible, and concrete, Applicants submit the Office Action apparently contradicts itself. On the one hand, the Office Action asserts the *practical application* of determining if a stall is due to loading of memory and flushing of an instruction after a predetermined number of cycles if data is to be loaded from memory before execution is not clear; on the other it asserts it is the *final result* achieved which *is useful, tangible, and concrete*, not the steps taken. Applicants submit this contradictory position is vitiated in that the practical application, the final result, and steps taken (*i.e.*, "determining", "flushing" – as discussed above) are all directed to a useful, tangible, and concrete result and supported by the specification. Applicants maintain the §101 rejection should be withdrawn.

Applicants respectfully submit that the cited references do not teach, suggest or describe "[a] method comprising: … flushing an instruction from said first thread from a pipeline of said processing system after a predetermined number of clock cycles if data is to be loaded from said memory device before executing said instruction" (*e.g.*, as described in claim 1).

The Office Action asserts Gottlieb teaches the relevant limitations, citing column 2, lines 54-61; column 7, lines 13-26; and column 1, lines 45-55. *See* Office Action dated 3/13/2007, paragraph 1, page 3. Applicants disagree. Column 2, lines 54-61 state:

> Coarse grained multi-threading can provide significant performance advantages when thread switches are triggered for those operations that would otherwise stall the processor's pipeline for an interval that is significantly longer than the time required to switch threads. An exemplary thread switching processes may require on the order of 30 cycles to flush or drain the instructions of the current thread from the pipeline, save the thread's architectural state information, and retrieve instructions from the newly scheduled thread.

The cited section is directed toward the alleged benefits of coarse grained multi-threading. It further approximates that an exemplary thread switch may take "on the order of" 30 cycles to flush or drain the current instructions from the pipeline, save the thread's state information, and retrieve instructions from a newly scheduled thread.

Applicants submit the cited section fails to teach or describe the relevant limitations. In short, performing a number of tasks A over an approximate of time A' is not the same as performing a task B after a certain, predetermined amount of time B' passes. In particular, the inconsistent, *ad hoc* completion of certain flushing or draining current instructions, saving a thread's state information, and retrieving instructions over a approximate, only exemplary, period of 30 cycles is not the same as flushing an instruction from a first thread from a pipeline of a processing system after a predetermined number of clock cycles if data is to be loaded from said memory device before executing said instruction. This cited section in Gottlieb does not describe flushing an instruction from a first thread from a pipeline of a processing system after a predetermined number of clock cycles as readily understood in the art and as described

in, for example, claim 1. Applicants submit the cited section fails to teach or describe the relevant limitations.

Column 7, lines 13-26 state:

> There has thus been provided a system and method for detecting long latency pipeline stalls in a processor. The present invention tracks the availability of data in specific registers and the status of bus requests to various memory structures. Circuitry is provided to map the registers to the bus requests and monitors the status of bus requests for indications that a long latency operation has been initiated. When a long latency operation such as a miss in the processor's caches, is detected, a register read queue is monitored to determine whether an instruction needs the data being accessed by the long latency operation. A thread switch condition is indicated when an instruction tries to access a register which is awaiting data from a long latency read operation.

The cited section describes the alleged benefits of the Gottlieb system with regard to long latency pipeline stalls. The cited section does not describe flushing an instruction at all, and therefore does not describe flushing an instruction from a first thread from a pipeline of a processing system after a predetermined number of clock cycles if data is to be loaded from said memory device before executing said instruction.

Column 1, lines 45-55 state:

> Switching processor resources from one thread to another may incur a performance penalty, since it takes time to flush or drain the pipeline of instruction from the current thread, save the thread's architectural state, and provide instructions from the new thread to the processor resources. These steps can take tens of clock cycles (on the order of 20 to 40 clock cycles) to complete. Coarse-grained multi-threading thus enhances performance only when the processor delay attributable to the thread switch condition is greater than the delay of the thread switching operation.

The cited section describes switching processor resources and the possible penalties involved. The Gottlieb reference estimates that flushing a pipeline, saving an thread's architectural state, and providing instructions from the new thread may take on the order of 20 to 40 clock cycles.

Similar to the section discussed above (*i.e.,* column 2, lines 54-61), Applicants submit the instant cited section only makes general estimations regarding the total time span required to perform various processor tasks (flushing, saving, providing instructions), of which one of the task is flushing. As maintained above, flushing or draining current instructions, saving a thread's state information, and retrieving instructions over a approximate period of 20 to 40 cycles is not the same as flushing an instruction from a first thread from a pipeline of a processing system after a predetermined number of clock cycles if data is to be loaded from said memory device before executing said instruction. Applicants submit the cited sections, and indeed the Gottlieb reference as a whole, fails to teach or describe the relevant limitations.

Next, the Office Action asserts that Eikemeyer teaches the relevant limitations at column 1, lines 6-12; column 6, lines 18-26, and column 5, lines 26-30. *See* Office Action dated 12/21/2005, paragraph 7. Applicants disagree.

Column 1, lines 6-12 state:

> The present invention relates in general to an improved data processing system and in particular to an improved system and method for switching threads of execution when execution of a thread is stalled in the dispatch stage of a multithread pipelined processor and flushing the stalled thread from earlier stages of pipeline.

The cited section describes that the Eikemeyer reference is directed to a method of switching threads of execution when a thread is stalled. It generally describes flushing a stalled thread from a pipeline, but does not describe flushing an instruction from a first thread from a pipeline of a processing system after a predetermined number of clock cycles if data is to be loaded from said memory device before executing said instruction.

Next, column 6, lines 18-26 state:

> …flush decode logic to determine if the thread having the stalled instruction has a previous flush condition, a dispatch flush mechanism to flush the thread having the stalled instruction from the fetch stage, the decode stage, and the dispatch stage if no other previous flush condition exists or if the previous flush condition has a lower priority than the stalled instruction so that the processor can process another of the independent threads of execution with the processor pipeline.

The cited section is directed to processor comprising flush decode logic to determine if there is a previous flush condition and a dispatch flush mechanism to flush a thread if the thread is stalled in one of a number of stages (*i.e.*, the flush stage, the decode stage) or determine whether a flush condition has a lower priority than a stalled instruction so that the processor can process other independent threads. It does not describe flushing an instruction from a first thread from a pipeline of a processing system after a predetermined number of clock cycles if data is to be loaded from said memory device before executing said instruction

Finally, column 5, lines 23-30 (including the cited lines 26-30) state:

> The method may also comprise restarting the thread whose instructions were flushed from the dispatch stages and all stages in the processor pipeline prior to the dispatch stage. The step of restarting may be delayed until a condition which causes the stalled instruction to stall at the dispatch stage is resolved. The step of restarting the flushed thread may also be delayed until a number of the processor cycles have passed.

The cited section discusses restarting a *previously flushed* thread. The reentry may be delayed until a condition causing the stall is resolved or until a number of processor cycles have passed. Again, similar to the cited section discussed above, Applicants submit this cited section is directed toward what can be done with a thread *after* it has been flushed, while the relevant limitation of claim 1 discussed above is clearly describes and is directed to, among other things, conditionals prior to a flush. Delaying the restart

of a previously flushed thread for a number of processor cycles is not the same as flushing an instruction from a first thread from a pipeline of a processing system after a predetermined number of clock cycles if data is to be loaded from said memory device before executing said instruction. Applicants submit this cited section fails to teach or suggest the relevant limitations as well.

Therefore, since each and every limitations is not found in the cited referenced, Applicants submit neither the Gottlieb nor Eickemeyer references can adequately form the basis of a proper 35 U.S.C. §102 rejections of independent claim 1. Independent claims 5, 10, and 16 contain substantively similar limitations and therefore are also allowable for similar reasons. Claims 2-4, 6-9, 11-15 and 17-21 depend from allowable independent claims 1, 5, 10 and 16, and therefore are in condition for allowance as well.

For at least the above reasons, Applicants respectfully submit that this application is in condition for allowance. A Notice of Allowance is earnestly solicited.

The Examiner is invited to contact the undersigned at (408) 975-7500 to discuss any matter concerning this application. The Office is hereby authorized to charge any additional fees or credit any overpayments to Deposit Account No. **11-0600**.

Respectfully submitted,

KENYON & KENYON LLP

Date: February 25, 2008

By: /Sumit Bhattacharya/_____
Sumit Bhattacharya
(Reg. No. 51,469)

KENYON & KENYON LLP
333 West San Carlos St., Suite 600
San Jose, CA 95110
Telephone: (408) 975-7500
Facsimile: (408) 975-7501

113049